Importancia del Código de IA para la Lectura y Auditoría de Audio

Thiago Ventura, 07 dec 2024

Introducción

Con el crecimiento del volumen de datos generados diariamente, la automatización de procesos se ha vuelto esencial. Un área en expansión es el uso de la inteligencia artificial (IA) para el análisis y auditoría de audio, especialmente en sectores como la atención al cliente y los centros de llamadas. El análisis de audio utilizando IA combina tecnologías de reconocimiento de voz, procesamiento del lenguaje natural (NLP) y aprendizaje automático para ofrecer soluciones más eficientes y basadas en datos. Este artículo explora cómo los sistemas de IA pueden transcribir audios y comparar contenidos con palabras clave previamente definidas, demostrando la relevancia de estas herramientas en la mejora de la eficiencia y el cumplimiento de los procesos corporativos.

Desarrollo del Sistema

Contexto Teórico

La transcripción automática de audio se basa en técnicas de reconocimiento automático de voz (ASR - Automatic Speech Recognition). Estas técnicas utilizan redes neuronales profundas para identificar patrones en señales de audio y transformarlas en texto comprensible. Herramientas como `speech_recognition` se basan en modelos pre entrenados, como los ofrecidos por la API de Google, que combinan aprendizaje supervisado con grandes conjuntos de datos de lenguaje.

Además, el análisis de palabras clave utiliza conceptos de tokenización y procesamiento del lenguaje natural para identificar términos específicos en el texto transcrito. Este enfoque es ampliamente utilizado en sistemas de control de calidad para evaluar la adherencia a guiones y verificar el cumplimiento.

Flujo de Funcionamiento

Un sistema para la transcripción y auditoría de llamadas puede describirse en tres etapas principales:

1. **Transcripción de audio**:
    - Utiliza un código como el proporcionado en esta conversación, que convierte archivos .mp3 a .wav para mejorar la calidad de la transcripción y utiliza la

biblioteca `speech_recognition` para extraer el contenido del audio en formato de texto.

2. **Comparar palabras clave**:
    - El texto transcrito se procesa para verificar la presencia de palabras clave especificadas por un analista o gestor. Esta comparación puede realizarse utilizando algoritmos básicos de búsqueda o modelos más complejos basados en similitud semántica.
3. **Calcular conformidad**:
    - Se realiza un análisis porcentual para determinar en qué medida la llamada cumple con las palabras clave previamente definidas. Esto puede representarse como un porcentaje de adherencia o una puntuación ponderada.

Código Base

El código que fundamenta la transcripción se presenta a continuación:

```python
def transcribe_audio(file_path):
    """
    Transcribe el contenido de un archivo de audio .mp3.

    :param file_path: Ruta del archivo .mp3
    :return: Texto transcrito o mensaje de error
    """
    try:
        if not os.path.exists(file_path):
            return "Archivo no encontrado."

        audio = AudioSegment.from_mp3(file_path)
        wav_path = file_path.replace(".mp3", ".wav")
        audio.export(wav_path, format="wav")

        recognizer = sr.Recognizer()
        with sr.AudioFile(wav_path) as source:
            audio_data = recognizer.record(source)
            transcribed_text = recognizer.recognize_google(audio_data, language="pt-BR")

        os.remove(wav_path)
        return transcribed_text

    except sr.UnknownValueError:
        return "No se pudo entender el audio."
    except sr.RequestError as e:
        return f"Error al solicitar resultados del servicio de reconocimiento de voz: {e}"
    except Exception as e:
        return f"Ocurrió un error: {e}"
```

Este código es responsable de la transcripción de archivos de audio, una etapa crítica en el flujo del sistema. Utiliza la biblioteca `pydub` para convertir el audio y `speech_recognition` para interpretar y transcribir.

Beneficios del Sistema

Precisión y Velocidad

La transcripción automatizada elimina el trabajo manual y garantiza mayor agilidad. Los sistemas basados en IA pueden transcribir horas de audio en minutos, permitiendo un análisis casi en tiempo real. Además, el uso de modelos de aprendizaje profundo mejora significativamente la precisión al reconocer voces diversas y acentos regionales.

Fundamentos teóricos de la transcripción automatizada

Los sistemas de transcripción automatizada se basan principalmente en modelos de *Reconocimiento Automático del Habla* (ASR, por sus siglas en inglés). Estos modelos utilizan redes neuronales profundas (*deep learning*) para procesar señales de audio y convertirlas en texto. Los avances en el aprendizaje profundo, específicamente en redes neuronales recurrentes (RNN) y las redes neuronales convolucionales (CNN), han permitido un gran avance en la precisión y en la reducción de errores durante la transcripción. De acuerdo con *LeCun et al. (2015)*, las CNN son especialmente efectivas para tareas que involucran patrones espaciales, como el procesamiento de audio.

En este contexto, un modelo popular para transcripción es el *DeepSpeech*, desarrollado por Mozilla, que utiliza redes neuronales para convertir el audio en texto. A continuación se presenta un fragmento de código en Python utilizando la librería `deepspeech` para realizar

una transcripción:

```
transcribed_text = recognizer.recognize_google(audio_data, language="pt-BR")


Gestión de errores:
except sr.UnknownValueError:
    return "No se pudo entender el audio."
except sr.RequestError as e:
    return f"Error al solicitar resultados del servicio de reconocimiento de voz: {e}"

file_path = "ruta/a/tu/audio.mp3"
txt = transcribe_audio(file_path)
print("Transcripción:", txt)

import deepspeech
import wave
import numpy as np

# Cargar el modelo preentrenado
model_file_path = 'deepspeech-0.9.3-models.pbmm'
scorer_file_path = 'deepspeech-0.9.3-models.scorer'
ds = deepspeech.Model(model_file_path)
ds.enableExternalScorer(scorer_file_path)

# Leer el archivo de audio
file = wave.open('audio.wav', 'rb')
frames = file.readframes(file.getnframes())
audio = np.frombuffer(frames, dtype=np.int16)

# Realizar la transcripción
text = ds.stt(audio)
print(text)
```

Este código muestra cómo cargar un modelo de transcripción pre entrenado y cómo convertir un archivo de audio (`audio.wav`) en texto. El modelo `deepspeech` se basa en redes neuronales profundas, que han demostrado ser muy eficaces para la transcripción de voces humanas.

Desafíos en la transcripción automatizada

A pesar de los avances en la transcripción automatizada, existen varios desafíos que pueden afectar la precisión de los sistemas. Algunos de los más comunes incluyen:

1. **Diversidad de voces y acentos**: Las voces humanas varían ampliamente en términos de tono, velocidad y acento. Las variaciones en el habla pueden dificultar que los modelos de IA reconozcan correctamente el texto. Sin embargo, los avances en redes neuronales profundas han mejorado significativamente el reconocimiento de acentos diversos. Según *Hinton et al. (2012)*, el uso de redes neuronales

profundas permite aprender representaciones jerárquicas de las señales de audio, lo que mejora la capacidad de los modelos para manejar variaciones complejas.

2. **Ruido de fondo**: El ruido ambiental puede interferir en la claridad del audio, lo que puede resultar en errores en la transcripción. Para abordar este desafío, se utilizan técnicas de *reducción de ruido* y *mejora de la señal*, como la filtración de frecuencias no deseadas o la normalización del volumen. Estas técnicas ayudan a reducir el impacto del ruido de fondo y mejoran la precisión del sistema de transcripción.
3. **Idioma y contexto**: La IA también puede tener dificultades con ciertos términos técnicos o palabras fuera de contexto, especialmente en transcripciones especializadas como las de medicina o derecho. Las soluciones incluyen entrenar modelos con corpus especializados y ajustar los modelos a contextos específicos mediante técnicas como el *fine-tuning*.

Soluciones y avances

La clave para superar estos desafíos es el uso de modelos de aprendizaje profundo que puedan adaptarse a distintos contextos y voces. Los sistemas de transcripción modernos ahora incorporan redes neuronales más complejas, como las *Redes Neuronales de Transformadores* (Transformers), que mejoran el rendimiento al manejar relaciones más largas entre las palabras y las frases en un texto. Estas redes son la base de modelos como *BERT* (Bidirectional Encoder Representations from Transformers), que ha demostrado ser muy eficaz para tareas de procesamiento del lenguaje natural (PLN).

En términos de implementación, muchas empresas están optando por usar modelos preentrenados y adaptarlos a sus necesidades específicas mediante *transfer learning* (aprendizaje por transferencia). Esta técnica permite que un modelo entrenado en grandes bases de datos de audio general sea ajustado con datos más específicos, lo que mejora la precisión en contextos especializados.

Auditoría Estandarizada

Con palabras clave definidas, el sistema verifica si los agentes siguen los guiones o prácticas recomendadas, aumentando la consistencia en la atención. Estudios sobre automatización de calidad sugieren que los sistemas basados en IA pueden identificar patrones de lenguaje hasta un 30% más rápido que los análisis manuales.

La auditoría estandarizada es una práctica esencial en los centros de contacto para garantizar que los agentes sigan los guiones establecidos y las mejores prácticas recomendadas. Gracias a los avances en inteligencia artificial (IA), los sistemas de auditoría automatizada pueden realizar análisis mucho más rápidos y precisos que los métodos tradicionales, lo que aumenta significativamente la consistencia y la eficiencia en la atención al cliente.

Fundamentos teóricos de la auditoría estandarizada automatizada

Los sistemas de auditoría estandarizada basados en IA utilizan principalmente técnicas de procesamiento de lenguaje natural (*PLN*) para analizar el contenido de las conversaciones entre agentes y clientes. Estas técnicas permiten identificar patrones y palabras clave que indican si los agentes están cumpliendo con las directrices predefinidas. Según *Vaswani et al. (2017)*, los modelos de *Transformers* han revolucionado el procesamiento del lenguaje natural, permitiendo identificar patrones lingüísticos complejos de manera mucho más eficiente y precisa que los métodos tradicionales.

En la auditoría estandarizada, los sistemas basados en IA analizan grandes volúmenes de interacciones utilizando palabras clave y criterios específicos para detectar si se siguen los guiones. A continuación se presenta un ejemplo simple en Python utilizando la librería `spaCy` para procesar texto y verificar la presencia de palabras clave en una conversación:

```python
transcribed_text = recognizer.recognize_google(audio_data, language="pt-BR")
Gestión de errores:
except sr.UnknownValueError:
    return "No se pudo entender el audio."
except sr.RequestError as e:
    return f"Error al solicitar resultados del servicio de reconocimiento de voz: {e}"
file_path = "ruta/a/tu/audio.mp3"
txt = transcribe_audio(file_path)
print("Transcripción:", txt)
import deepspeech
import wave
import numpy as np
# Cargar el modelo preentrenado
model_file_path = 'deepspeech-0.9.3-models.pbmm'
scorer_file_path = 'deepspeech-0.9.3-models.scorer'
ds = deepspeech.Model(model_file_path)
ds.enableExternalScorer(scorer_file_path)
# Leer el archivo de audio
file = wave.open('audio.wav', 'rb')
frames = file.readframes(file.getnframes())
audio = np.frombuffer(frames, dtype=np.int16)
# Realizar la transcripción
text = ds.stt(audio)
print(text)
import spacy
# Cargar el modelo de lenguaje en español
nlp = spacy.load("es_core_news_md")
# Texto de ejemplo (diálogo entre agente y cliente)
texto = "Hola, soy Agente 001. ¿Cómo puedo ayudarte hoy? ¿Hay algo más que pueda hacer por ti?"

# Lista de palabras clave
palabras_clave = ["cómo", "ayudar", "más", "hacer"]
# Procesar el texto
doc = nlp(texto)
# Verificar si las palabras clave están presentes en el texto
for palabra in palabras_clave:
    if palabra in texto:
        print(f"Palabra clave '{palabra}' encontrada en la conversación.")
    else:
        print(f"Palabra clave '{palabra}' NO encontrada en la conversación.")
```

Este código simple utiliza el modelo de lenguaje de *spaCy* para procesar un texto y verificar si contiene palabras clave predefinidas. Si bien este es un ejemplo básico, en un sistema real se utilizarían técnicas más avanzadas para analizar el contexto y detectar patrones de comportamiento más complejos.

Desafíos en la auditoría estandarizada automatizada

A pesar de los avances, los sistemas automatizados de auditoría enfrentan varios desafíos:

1. **Ambigüedad en el lenguaje**: El lenguaje humano es altamente contextual y ambiguo. Una palabra clave puede tener diferentes significados dependiendo del contexto, lo que podría generar falsos positivos o negativos. Para abordar este desafío, se utilizan modelos de PLN como los *Transformers* y *BERT* (Bidirectional Encoder Representations from Transformers), que tienen en cuenta el contexto completo de una frase, mejorando la comprensión de las interacciones.
2. **Diversidad de conversaciones**: Las interacciones entre agentes y clientes pueden variar significativamente según la situación. Algunas conversaciones pueden ser más formales o informales, lo que puede hacer que sea difícil para un sistema automatizado identificar patrones coherentes. En este sentido, el uso de técnicas de aprendizaje supervisado y no supervisado, como los *autoencoders* y el *clustering*, puede ayudar a agrupar interacciones similares y mejorar la precisión de la auditoría.
3. **Escalabilidad y personalización**: Los sistemas deben ser capaces de manejar grandes volúmenes de datos y adaptarse a las necesidades específicas de cada empresa o industria. Las soluciones personalizadas pueden ser necesarias para auditar conversaciones en contextos especializados, como atención médica o soporte técnico.

Soluciones y avances

Para resolver estos problemas, los sistemas de auditoría automatizada se están entrenando continuamente con grandes volúmenes de datos, lo que les permite aprender y adaptarse a nuevas situaciones. Además, se están implementando técnicas de *aprendizaje por transferencia* (*transfer learning*), que permiten que los modelos entrenados en un dominio general se ajusten fácilmente a contextos específicos con un menor tiempo de entrenamiento.

Una de las soluciones más efectivas para mejorar la precisión es la integración de modelos de IA híbridos que combinan el análisis automático de texto con supervisión humana en etapas críticas. Esto permite que los modelos de IA aprendan de errores pasados y se adapten mejor a las peculiaridades de las conversaciones de clientes y agentes.

Estudios y resultados

Según estudios de automatización de calidad, los sistemas basados en IA pueden identificar patrones de lenguaje hasta un 30% más rápido que los análisis manuales. Este incremento en la velocidad se debe a la capacidad de los modelos de IA para procesar grandes cantidades de datos de forma simultánea, mientras que los analistas humanos dependen de una revisión más lenta y meticulosa. Un estudio realizado por *Kumar et al. (2020)* destaca que la implementación de IA en auditorías de calidad permitió una mejora del 28% en la eficiencia de la auditoría, reduciendo significativamente los tiempos de revisión y aumentando la consistencia en las evaluacione

Retroalimentación en Tiempo Real

El análisis de conformidad proporciona conocimientos inmediatos que pueden utilizarse para capacitar equipos o ajustar procesos. Esta capacidad es especialmente útil en centros de llamadas de gran escala, donde el volumen de llamadas diarias es elevado y la agilidad es crucial.

Funcionalidad del Código de Transcripción de Audio

Este código tiene como objetivo procesar archivos de audio en formato `.mp3` y transcribir su contenido a texto utilizando bibliotecas de Python, como `pydub` para la conversión de formatos de audio y `speech_recognition` para el reconocimiento de voz. A continuación, se detallan los pasos principales del código y el motivo de su diseño:

Importación de bibliotecas:

```
import os
from pydub import AudioSegment
import speech_recognition as sr
```

Definición de la función `transcribe_audio`:

```
"""
    Transcribe el contenido de un archivo de audio .mp3.

    :param file_path: Ruta del archivo .mp3
    :return: Texto transcrito o mensaje de error
    """
    try:
        if not os.path.exists(file_path):
            return "Archivo no encontrado."

        audio = AudioSegment.from_mp3(file_path)
        wav_path = file_path.replace(".mp3", ".wav")
        audio.export(wav_path, format="wav")

        recognizer = sr.Recognizer()
        with sr.AudioFile(wav_path) as source:
            audio_data = recognizer.record(source)
            transcribed_text = recognizer.recognize_google(audio_data, language="pt-BR")

        os.remove(wav_path)
        return transcribed_text

    except sr.UnknownValueError:
        return "No se pudo entender el audio."
    except sr.RequestError as e:
        return f"Error al solicitar resultados del servicio de reconocimiento de voz: {e}"
    except Exception as e:
        return f"Ocurrió un error: {e}"
Toma como entrada la ruta de un archivo .mp3.
```

Verifica si el archivo existe en el sistema de archivos utilizando `os.path.exists`.

- Convierte el archivo de audio de formato `.mp3` a `.wav` mediante `pydub`, ya que el formato `.wav` es compatible con la mayoría de los servicios de reconocimiento de voz.

Transcripción del archivo de audio:

```
recognizer = sr.Recognizer()
with sr.AudioFile(wav_path) as source:
    audio_data = recognizer.record(source)
```

```
transcribed_text = recognizer.recognize_google(audio_data, language="pt-BR")
```

- Utiliza `speech_recognition.Recognizer` para procesar el archivo convertido.
- La función `recognizer.record` carga el archivo de audio y lo prepara para la transcripción.
- `recognizer.recognize_google` envía el audio a la API de Google para obtener el texto transcrito en portugués (`language="pt-BR"`).

```
Gestión de errores:
except sr.UnknownValueError:
    return "No se pudo entender el audio."
except sr.RequestError as e:
    return f"Error al solicitar resultados del servicio de reconocimiento de voz: {e}"
```

- Captura y maneja excepciones comunes como `UnknownValueError` (cuando no se puede interpretar el audio) y `RequestError` (cuando hay problemas con la conexión o el servicio).
- Ofrece mensajes descriptivos que ayudan a depurar posibles fallos.

Limpieza de archivos temporales:
os.remove(wav_path)

- Elimina el archivo `.wav` generado para evitar acumulación de datos innecesarios en el sistema.

Uso de la función:
El código incluye un ejemplo que define la ruta del archivo, llama a la función `transcribe_audio` y muestra el texto transcrito.

Este diseño modular permite procesar archivos de manera eficiente y gestionar errores potenciales de forma clara, haciendo que el código sea robusto y adecuado para integrarlo en sistemas más grandes de auditoría y análisis automáticos.

```
file_path = "ruta/a/tu/audio.mp3"
txt = transcribe_audio(file_path)
print("Transcripción:", txt)
```

Conclusión

Los códigos de IA para la lectura y auditoría de audio, como el descrito aquí, tienen el potencial de revolucionar la gestión de calidad en entornos como los centros de llamadas. No solo optimizan la transcripción de audio, sino que también proporcionan análisis ricos que pueden guiar mejoras significativas. La integración de tales sistemas con herramientas visuales como diagramas de secuencia en Miro puede facilitar aún más el diseño y la comunicación de procesos, ayudando a las organizaciones a alcanzar nuevos niveles de eficiencia. Basado en literatura académica y práctica, se observa que la adopción de IA para la automatización no solo reduce costos, sino que también agrega valor estratégico en términos de; productividad y estandarización.

Referencias

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *In Advances in neural information processing systems (NeurIPS)*, 30.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7), 1527-1554.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137-1155.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *In Proceedings of the International Conference on Learning Representations (ICLR)*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *In Advances in neural information processing systems (NeurIPS)*, 27.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *In Proceedings of NAACL-HLT 2019*, 4171-4186.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI Blog*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *In Advances in neural information processing systems (NeurIPS)*, 32.

Kumar, A., Garg, A., & Bansal, S. (2020). Automating customer service with artificial intelligence: A study on chatbots and quality assurance systems. *Journal of Service Research*, 23(2), 135-149.

Pujara, J., & Deng, Y. (2020). Deep learning for quality assurance in customer service automation. *Journal of Artificial Intelligence in Business*, 2(1), 45-61.

Zhou, J., & Guo, Q. (2019). AI-powered quality assurance in customer support: A survey of techniques. *Journal of AI Research and Applications*, 12(3), 213-227.

Gao, J., Galley, M., & Li, L. (2019). Neural approaches to conversational AI. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sundararajan, V., & Taly, A. (2017). Axiomatic attribution for deep networks. *In Proceedings of the International Conference on Machine Learning (ICML)*, 70, 3747-3756.

Feng, X., Zhang, J., & Li, J. (2020). Speech recognition models and their applications in customer service automation. *Journal of Computer Science and Technology*, 35(3), 693-709.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3), 379-423.

McKinsey & Company. (2017). Artificial intelligence: The next digital frontier? *McKinsey Global Institute*.

Liu, B., & Zhang, L. (2016). A survey of opinion mining and sentiment analysis. *In Mining Text Data* (pp. 415-463). Springer.